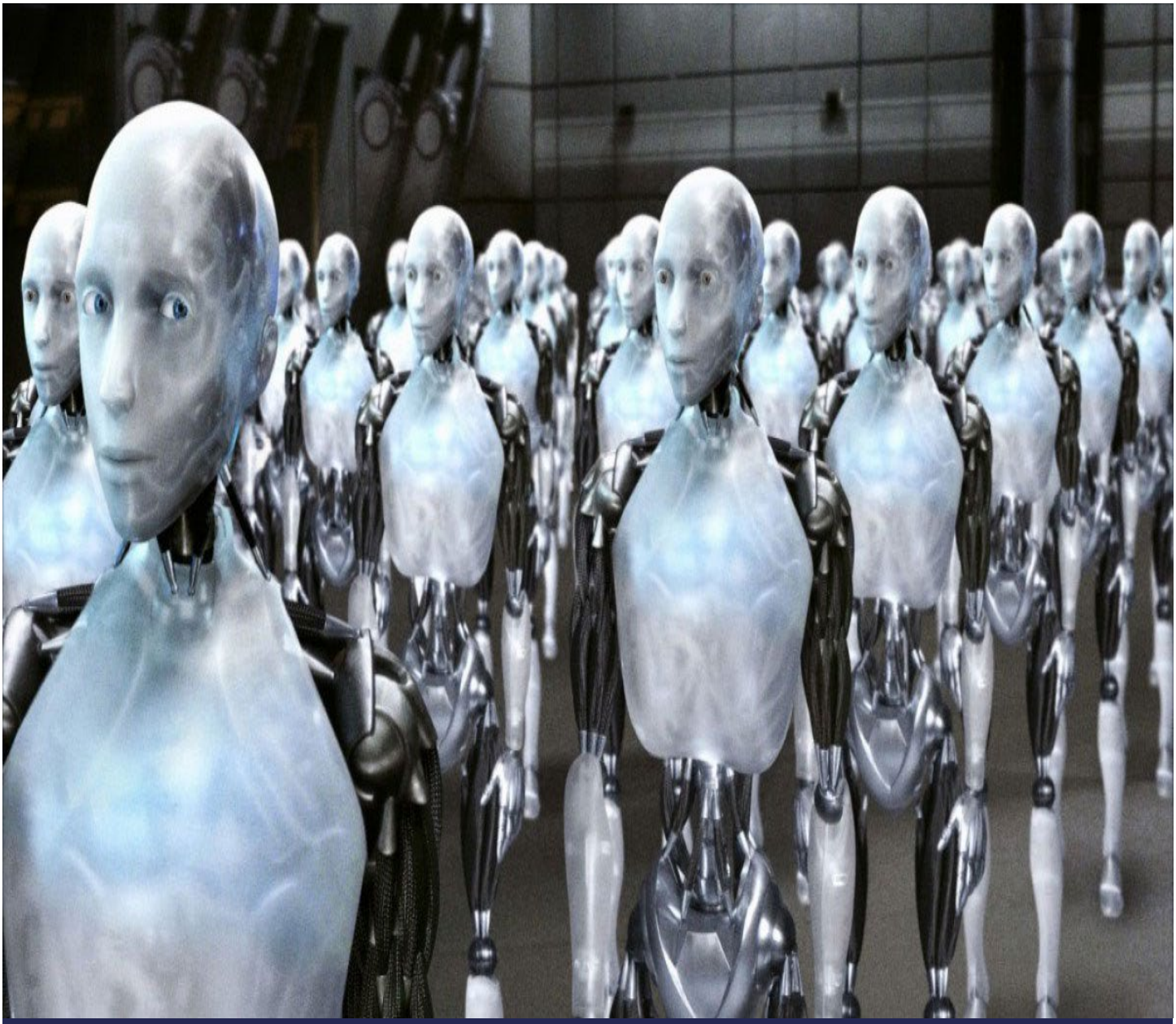


Assignment 2 Group Submission (iRobot)

Noor Ul Ain Khurshid (102763334), Saad Sultan (101227910), Alec Vince Gonzales Abuan (101220553)

Declaration and Statement of Authorship

1. I/we have not impersonated, or allowed myself/ourselves to be impersonated by any person for the purposes of this assessment.
2. This assessment is my/our original work and no part of it has been copied from any other source except where due acknowledgement is made.
3. No part of this assessment has been written for me/us by any other person except where such collaboration has been authorised by the lecturer/teacher concerned.
4. I/we have not previously submitted this work for this or any other course/unit.
5. I/we give permission for my/our assessment response to be reproduced, communicated, compared and archived for plagiarism detection, benchmarking or educational purposes.

I/we understand that:

- Plagiarism is the presentation of another person's work as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to exclusion from the university.
- Plagiarised material may be drawn from published and unpublished written documents, interpretations, computer software, designs, music, sounds, images, photographs, and ideas or ideological frameworks gained through working with another person or in a group.
- Plagiarised material can be drawn from, and presented in, written, graphic and visual form, including electronic data and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited.

I/We agree and acknowledge that:

1. I/we have read and understood the Declaration and Statement of Authorship above.
2. I/we accept that use of my/our Swinburne account to electronically submit this assessment constitutes my/our agreement to the Declaration and Statement of Authorship.
3. If I/we do not agree to the Declaration and Statement of Authorship in this context, the assessment outcome may not be valid for assessment purposes and may not be included in my/our aggregate score for this unit.

Penalties for plagiarism range from a formal caution to expulsion from the university, and are detailed in the Plagiarism and Misconduct webpage.

Who Did What?

Introduction to Artificial Intelligence Assignment 2 Group Submission	
Noor Ul Ain Khurshid	Created the slides from the report Co-developed the Convolutional Neural Network
Saad Sultan	Helped Write the report and create slides, participated in video Developed Random Forest algorithm classification
Alec Vince Gonzales Abuan	Helped Write the report Co-developed the Convolutional Neural Network.

We declare this is an accurate description of team contributions of the team members

Team Member Name	Signature	Date
Noor Ul Ain Khurshid	Noor Ul Ain Khurshid	23/5/22
Saad Sultan	Saad Sultan	23/5/22
Alec Vince Gonzales Abuan	Alec Vince Gonzales Abuan	23/5/22

Contents

Video Submission

Link here -> https://www.youtube.com/watch?v=4JXVPJOW4mA

Data preparation and pre-processing

Figure 1: Dataset folders

○ What dataset have you chosen apart from the given datasets? Why?

The chosen dataset was a dataset for garbage collection from the website Kaggle. It must be said that this dataset will be used for both the shallow and deep learning models. The dataset was a 12-class collection of photos containing pictures of garbage. The classes indicated were a good mixture of recyclable and non-recyclable garbage such as battery, biological, brown-glass, cardboard, clothes, green-glass, metal, paper, plastic, shoes, trash, and white-glass. Additionally, this data set was intended by the author for multi-class image classification therefore it can be useful if rearranged for our purpose of binary classification. To add more information regarding the dataset, the ideal scenario intended by the author was to create a set of images which would mimic a scenario where a camera is setup on top of a garbage conveyor to capture photos for processing. This makes this set of data a good candidate for real time classification of garbage with a practical use. The images are selected to mimic those specific conditions and often have a clear background in comparison.

In comparison to other datasets, there are multiple conditions why this dataset was chosen over others. Relevancy of required elements in a data set is the first factor which is considered in selecting a good dataset (*Find Good Data Sets* 2022). Extensive content with the elements needed to create the model is the first strength of this acquired image dataset. The dataset has the specified sets of images with variety of classes which will aid the process of training and testing for this binary classification. During the training and testing of the model, an additional requirement that must be fulfilled which is a common problem in finding datasets is the quantity of images or data (*How to Ensure Image Dataset Quality for Image Classification?* 2022). Therefore, this dataset of images is more than enough for this purpose having 12 classes sharing a total 15,150 images which are divided into training and testing sets making around 12,120 images for training and 3030 images for testing. In general, according to Picsellia, a good rule of thumb to follow is to have a minimum 1000 images per class (*How to Ensure Image Dataset Quality for Image Classification?* 2022).

Therefore, since this model is built for a binary classification, it was plenty enough for the process. Another factor to consider in a dataset of images is the fault of mislabelling of images (*How to Ensure Image Dataset Quality for Image Classification?* 2022). In this dataset, the images are well labelled and organised to their respective pre-determined classes and are none to minimal fault to mislabelling. Finally, a best practice in finding a dataset is to consider challenges and best practice of selection images considering the photos having certain features such as variable viewpoints, variable scales, visibility differences and lightning conditions (*How to Build a Dataset for Image Classification* 2022). Admittedly, some of the factors considered does not fully support the chosen dataset but satisfies most of them on at least a minimal level.

- ## How many classes? How did you obtain it? How was it processed?

Initially, the dataset was organised and labelled into 12 classes which were battery, biological, brown-glass, cardboard, clothes, green-glass, metal, paper, plastic, shoes, trash, and white-glass. All these folders contain respective photos each with the name of the folder with numbers till the total amount of the folder. The classes are also varying in amounts per folder with the shoes and the clothes folders having the most amounts while the brown-glass and green-glass having the least amount. The dataset was acquired kindly thru the website Kaggle from an author. The author initially had the intention to collect data to mimic a setup of having a camera on top of a garbage disposal taking pictures. This results to images which are close-ups with often a distinctive background and content separation.

The processing of the image dataset can be separated into parts. The firs processing stage was done by manual labour of combining the 12 classes into classes needed for the binary classification which are needed by the models. Thru a website reference, we manually checked which of the 12 classes are considered recyclable and non-recyclable. After that, we then added them the photos of each respective classes into the right folder of either recyclable or non-recyclable. The classes chosen for recyclables were battery, brown-glass, cardboard, green-glass, white-glass, metal, paper, and plastic. On other side, we had chosen biological, clothes, shoes and trash to be non-recyclables. After we had clearly separated the binary classification image sets, we proceeded to portion the images into train and test data with the ration of 80% for training and 20% for testing. We had ensured to include a significant amount for training to ensure the most optimal result. During this process, we also considered to add validation set but opted out of it. We had a reach a conclusion that further dividing the dataset especially the training dataset to a lot for validation will only result to inaccuracies to the model due to the smaller number of images we have.

The next stage was done thru the Jupyter notebook before proceeding into the later stages of the model development. We had dedicated a section for pre-processing data to ensure the most optimal results possible. Using the python code, we had use the TensorFlow keras library ImageDataGenerator to prepare the image set for the model. Although it is true this step is considered pre-processing, it must be explained that Keras ImageDataGenerator function contrary to popular beliefs it is not an additive to the current dataset of images which will be used in the model training (Rosebrock 2019). Using this function, we are instead replacing the actual original batch of image dataset with the new randomly transformed batch of the original pictures with the given parameters. In our case, we had opted for certain image transforming parameters such as rescaling to the images to a ratio of 1 to 255, changing the slant to a range of 0.2, adding a zoom range of 0.2, and as well as flipping the image horizontally. It is also noted that we had done extensive parameters for the training data but only applied rescaling from 1 to 255 to the test data.

○ What are the potential applications that can be created from applying machine learning to this dataset?

The dataset was initially created to simulate a camera live feed on conveyor taking photos for an AI application of multiclass classification. The first application can be used to train a model for a garbage disposal to segment the trash into multiclass classification of trash as intended. As we find out in our research, it is incredibly useful to know what exactly the trash is. In this case, the multiclass identification can help further develop a company into segmenting recyclable and nom-recyclable materials. Recycling materials is not a binary classification and require a more complex system since there further constraints even within each class.

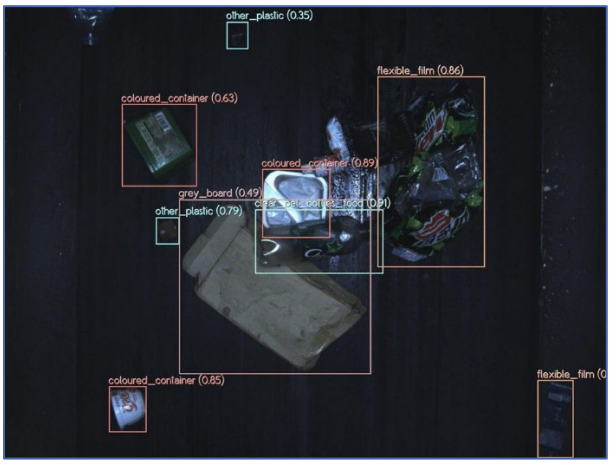


Figure 2: Greyparrot AI analysis

A good example for this application would be the Greyparrot project of creating an AI-powered computer software that aids in automation and transparency of recycling (Hackl 2020). Another possible use of this dataset is in the use of intelligent garbage bins. With the help of IoT and smart garbage bins with sensors, we can use an AI trained with this dataset to optimise the process (Michael 2021). With the help of AI and computer vision, we can use machine learning to allow sensors to classify trash as they are filled (Michael 2021).

Figure 3: Bin.e smart waste bin

A good example of this project is by Bin.e, using machine learning they classify and sort the waste immediately after they are thrown away (Michael 2021). Additionally, an application can also be used to let users know regarding the status of the bin and if they are overflowing show the direction to the nearest available bin to discourage overflowing (Michael 2021).

Training

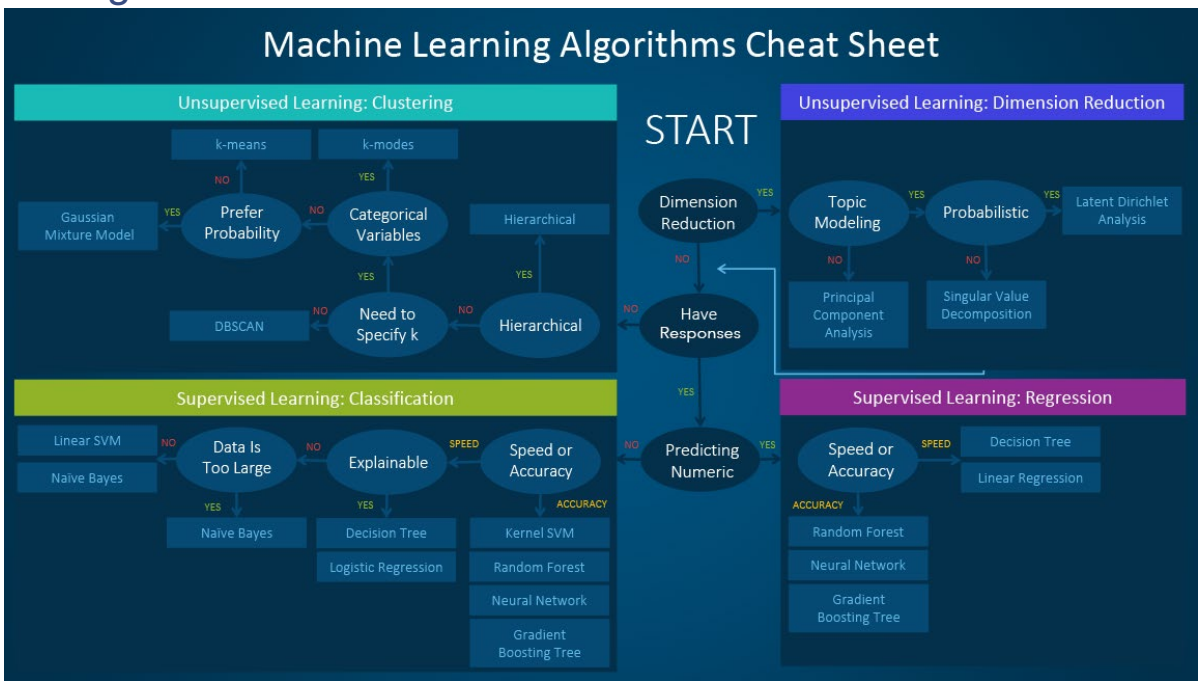


Figure 4: Machine learning cheat sheet

- What type of machine learning algorithms can be used in your program?

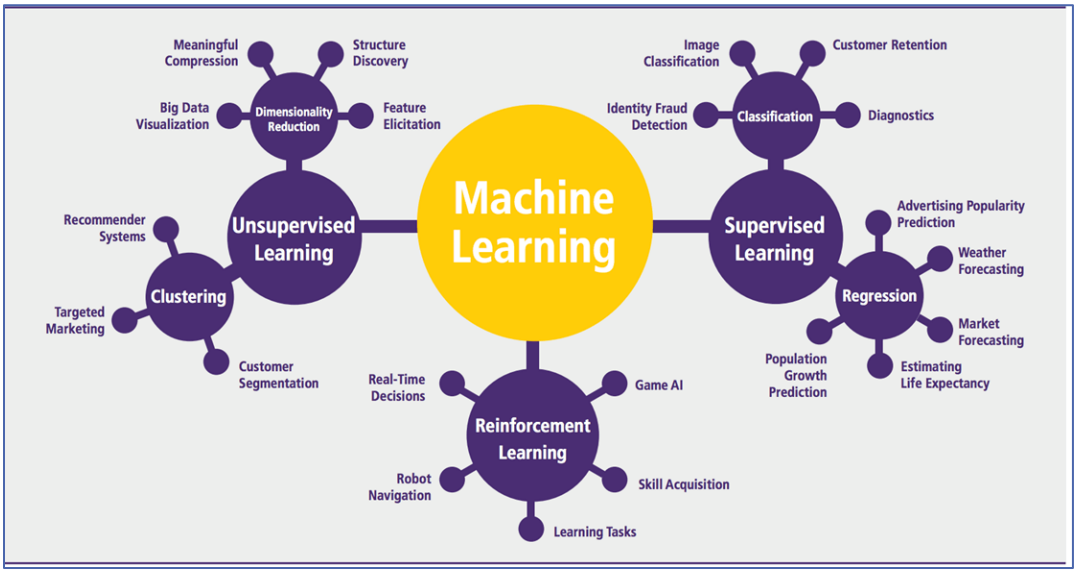


Figure 5:Machine Learning Types

Machine learning can be described and referred to as analysis of predicament or models of prediction (Wakefield 2022?). Predicting an output within an acceptable range, is the heart of machine learning and this is done by using programmed algorithms receiving and analysing input data (Wakefield 2022?). To improve the performance, these 'machines' are fed with more new data improving their 'intelligence' by learning and further optimisation (Wakefield 2022?). In total there are 4 types of machine learning algorithms, which are: Supervised learning, semi-supervised

learning, unsupervised and reinforcement learning. In general, we can use each of the different types to this project each differing in process, complexity, and goals.

- How do these algorithms work?

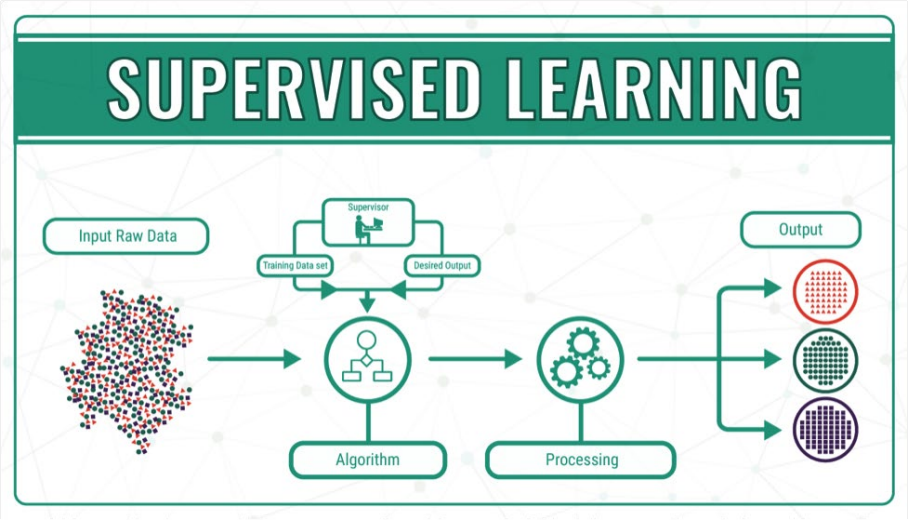


Figure 6: Supervised learning

The first type of algorithm that can be used is Supervised learning. In general, it can easily be described as machines learning thru human teaching by examples (Wakefield 2022?). Within this learning, a human operator is tasked to provide the algorithm with a labelled dataset that has the data including both wanted and unwanted inputs and outputs (Wakefield 2022?). Next, the algorithm provided with the dataset is tasked to conclude a method to reach those desired inputs and outputs (Wakefield 2022?). With the aid of the operator's knowledge, the algorithm is learning the dataset using observations, learning the data patterns and then finally creating predictions (Wakefield 2022?). After the predictions, the operator evaluates the correctness and then make changes to correct the algorithm and continuing this cycle until a desired high level of accuracy is acquired (Wakefield 2022?).

Additionally, under supervised machine learning there are 3 main tasks which are classification, regression, and forecasting (Wakefield 2022?). Classification is described as algorithms drawing conclusions from observations of data onto new sets of data and figuring out what category they belong to (Wakefield 2022?). A good example of this is an algorithm used to filter if emails are spam or not spam emails. Additionally, this project task falls under this category

Furthermore, under supervised machine learning there are 3 main tasks which are classification, regression, and forecasting (Wakefield 2022?). Classification is described as algorithms drawing conclusions from observations of data onto new sets of data and figuring out what category they belong to (Wakefield 2022?). A good example of this is an algorithm used to filter if emails are spam or not spam emails (Wakefield 2022?). Additionally, this project task falls under this category requiring to binary classify images if they are recyclable or not. Another type of task is regression. In this task, the algorithm is required to understand the association between variables and then make estimations on new sets of data (Wakefield 2022?). The learning needs the algorithm to recognize and focus on one independent variables and multiple/one changing variables proving incredibly useful for predictions and forecasting (Wakefield 2022?). Lastly, the third task is forecasting. In

this task, we use the algorithm to learn existing data and their history and trends to make predictions and analyse trends (Wakefield 2022?).

For this project, we opted to use supervised learning for ease of use and to cement our learning by taking it step by step.

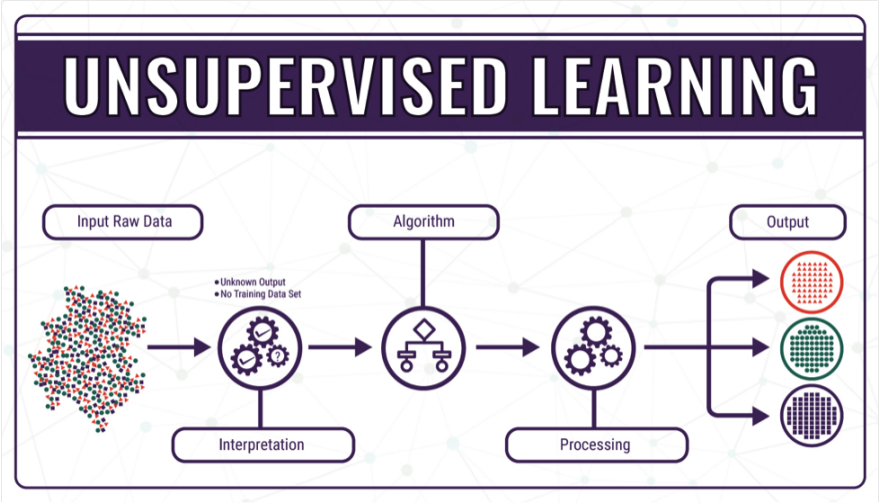


Figure 7: Unsupervised learning

The easiest explanation for unsupervised learning describes the process as an algorithm that studies data and identify patterns (Wakefield 2022?). In this learning method, no human interaction is required in providing a key or instructions (Wakefield 2022?). Additionally, the algorithm is isolated to learn and interpret large data (Wakefield 2022?). During this process, the algorithm will try different means of organising the data set to create a description of its structure (Wakefield 2022?). Different methodologies could be used such as clustering or arrangement of data into a more organised manner (Wakefield 2022?). As the learning continues over time, more data are assessed by the algorithm increasing its judgement ability to be more refined and improved (Wakefield 2022?).

In this learning method, there 2 tasks which falls under it: Clustering and dimension reduction (Wakefield 2022?). Clustering is the task of grouping sets of similar data into groups (Wakefield 2022?). It has many uses such as segmentation of data into groups and analysis of data sets for pattern recognitions (Wakefield 2022?). The other task is dimension reduction. Focusing on reduction, it decreases the variable amounts that are considered to pin-point the information needed (Wakefield 2022?).

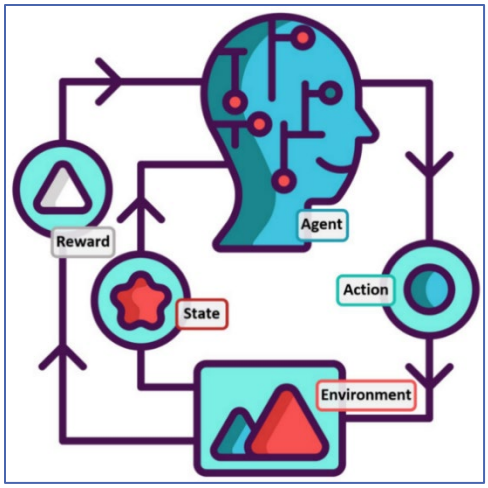


Figure 8: Reinforcement learning

Reinforcement learning is a type of machine learning where the process is strictly controlled to find the most optimal method to the best possible end values (Wakefield 2022?). In this algorithm, the operator sets the conditions such as set of actions, parameters, and end values/goals (Wakefield 2022?). With a defined environment, the algorithm is then free to explore different options and possibilities while monitoring and evaluating each result and concluding to the most optimal options (Wakefield 2022?). The method uses trial and error as a mean of teaching thru a set of given parameters (Wakefield 2022?). The algorithm improves by learning its past actions and adapting its approach to the situation to find the best possible result (Wakefield 2022?).

- ## What classifiers have you chosen?

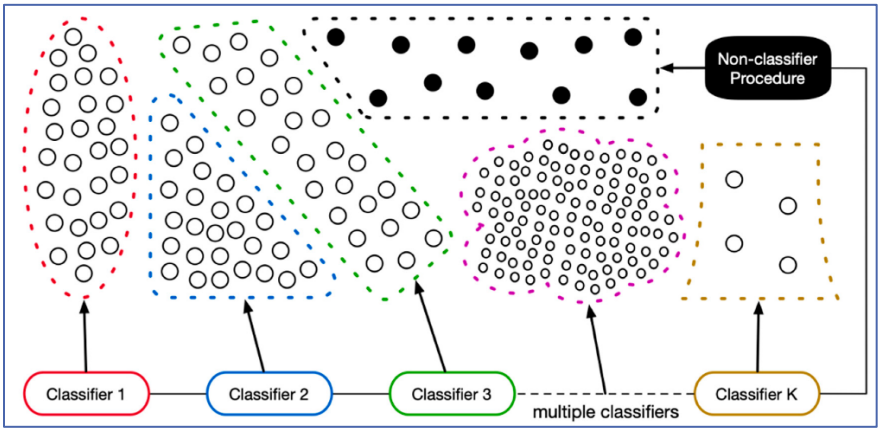


Figure 9: Classifiers

In machine learning, classifiers are defined as an algorithm which orders or sort data automatically into single or multiple set of classes (Mesevage 2020). Additionally, it can be said that the classifier is the algorithm itself used by the machine to learn and classify the data (Mesevage 2020).

As the assignment requires a deep machine learning algorithm. The first classifier we have chosen is an artificial neural network – Convolutional Neural Network. Artificial neural network is described as a collection of algorithms working together to solve problems that function like a human brain (Mesevage 2020). Connecting problem-solving processes, they chain together by

processing the result of each other by working sequentially (Mesevage 2020). Once one algorithm or problem has been solved, the next algorithm is then begun using the given result (Mesevage 2020). Artificial neural networks require larger datasets for training data to process and properly learn but have the edge on better performance in comparison to other algorithms (Mesevage 2020).

On the other hand, for our non-deep learning algorithm which is a more traditional and broader version of machine learning, the random forest classifier was chosen. This classifier includes the use of multiple decision trees to classify an image. The number of trees depends on the number of n_estimators and increase the accuracy of the classifier with each tree. The way this works is that the Random Forest Classifier gets the best result by getting an average of decision trees's results by using a voting system. However, too many trees can sometimes make the procedure more complex and thus the optimal number of trees was reached by trial and error to achieve the best accuracy. In our case, 50 decision trees were utilised to strengthen the training procedure as much as possible.

What is the classifier training procedure?

Deep Learning

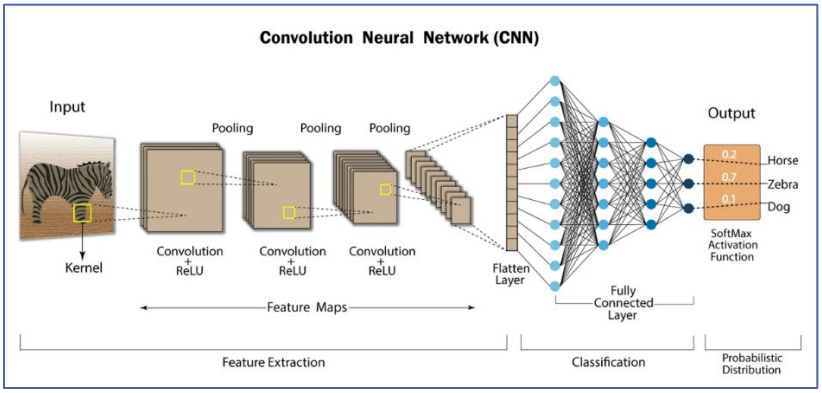


Figure 10: Convolutional neural network

For the artificial neural network, we are using convolutional neural networks. Made up of neurons with learnable weights, Neural networks have neurons in the system that receives inputs, takes in a sum of weights in which it then passes it through the activation function and respond with an output (Simran Bansari 2019). CNN have become popular in the world of image processing such as classifying, clustering and object recognition (Simran Bansari 2019). With this in mind, we have chosen CNN as it is useful for image processing in our project for deep learning.

CNN has four main layers which affects the training procedure (*Working of Convolutional Neural Network - Javatpoint* 2021). The four layers are: Convolutional, ReLU, Pooling and Fully Connected later (*Working of Convolutional Neural Network - Javatpoint* 2021).

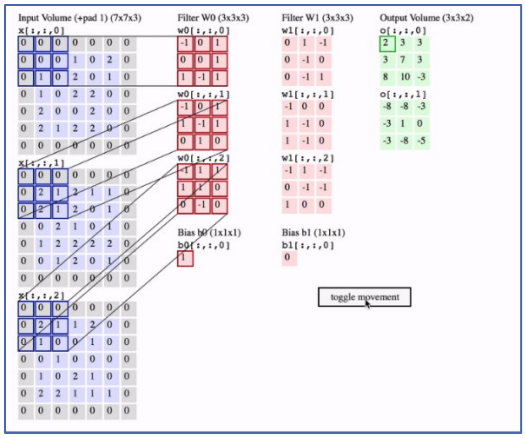


Figure 11: Convolutional layer

The first layer is the Convolutional layer which make this neural network a CNN and essentially its core building block that does majority of computing of the data (Simran Bansari 2019). Using filters or kernels, the data or image are convolved throughout the process (Simran Bansari 2019). With the use of a sliding window, we can describe filters as small units that we apply to our data or images. The process involves adding the specific values of the sliding action of the filter as it takes the element-wise product of filters in the image. The output of a convolutional layer for an image with a 3d filter with colour is a 2d matrix (Simran Bansari 2019). As images are made of pixels ranging from 0 to 255, we can use a specific size filter like 3x3 or any size filters with random generated numbers. With this filter, we can then calculate using the filter starting from the top left and then striding by 1 or any specific amount and get the product of each pixel multiplying with its counter part in the filter and once all are multiplied, we add the sum together dividing it by the total area of the filter or 9 in this case. The sliding filter is also known as kernel or neuron while the number inside them are referred to as weights or parameters and the region it occupies is called receptive field (Simran Bansari 2019).

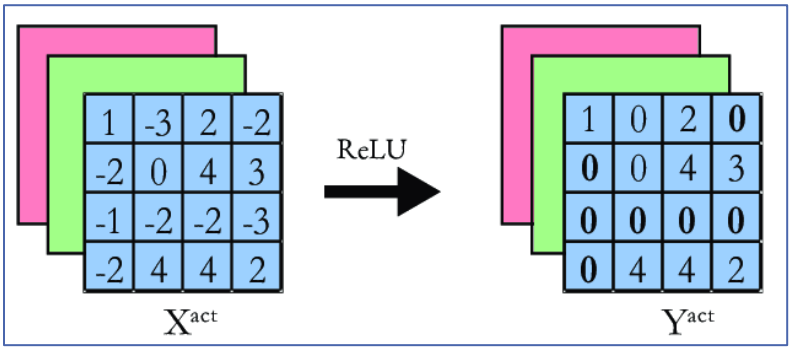


Figure 12: ReLU layer (Rectified Linear Unit)

The second layer is the activation layer which uses ReLU (Rectified Linear Unit) (Simran Bansari 2019). In this layer, we are using a rectifier function to increase non-linearity in the CNN as images are often a collection of differing objects not linear to one another (Simran Bansari 2019). In simple terms of an example, often we remove negative in favour of positive values as it gives benefits such as prevention of exponential growing of computing power requirements in the neural network (Simran Bansari 2019). It also helps feature the traits better the filter is looking for allowing for better models (Simran Bansari 2019).

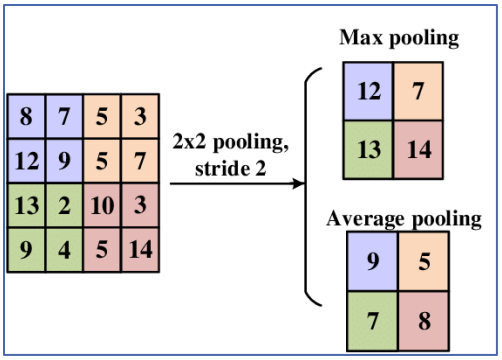


Figure 13: Pooling layer

The third layer is the pooling later (Simran Bansari 2019). In this layer, we are down sampling features (Simran Bansari 2019). There are two parameters that kept in mind in this layer: dimension of spatial extent and stride (Simran Bansari 2019). Dimensional spatial extent is concerned about the value of n that we take N across to represent a feature and map to a single amount (Simran Bansari 2019). The other is stride which is about the quantity of features the sliding window will skip across along the width and weight (Simran Bansari 2019). There are also different types of pooling such as max pooling filter and average pooling filter (Simran Bansari 2019). The max filter will return the highest value in the features while average filter will return the average value in the features (Simran Bansari 2019).

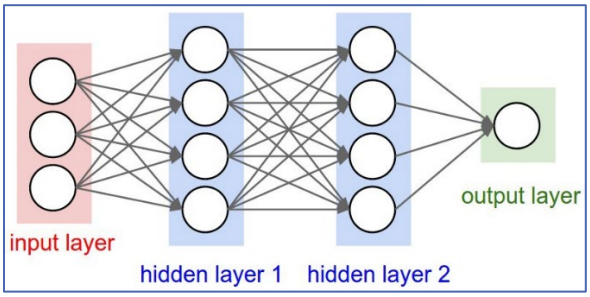


Figure 14: Activation layer

The final layer is the fully connected layer (Simran Bansari 2019). The layer is concerned about flattening by the transformation of the whole feature map matrix into one column inputted to the neural network for processing (Simran Bansari 2019). Adding the fully connected layer, the features are now combined to create a model which is then followed by an activation function such as SoftMax or sigmoid for classification of the output (Simran Bansari 2019).

Traditional Machine Learning

Unlike in Deep Learning, the training procedure for this algorithm requires a step-by-step approach in that feature extraction and classification are separate with the former being performed first. Images are first processed by being resized to 128 * 128 with the labels being encoded to be able to be understood by the computer. Once the image processing is completed, the feature_extraction() method is used to extract all the features of each and every training image from the training directory. The two main aspects of the feature extraction process are first acquiring the pixel values of the image as well using gabor filters to extract as much unique information about an image as possible. A total of 3200 images were used to extract the features with half part of the recyclable class while the other half represented the non-recyclable label.

Pixel values are one of the most important type of features that can be extracted from an image. As it reveals different attributes of the picture such as it level of brightness indicating whether its bright or dark. For this reason, the image was reshaped to one vector to access all the pixel values.

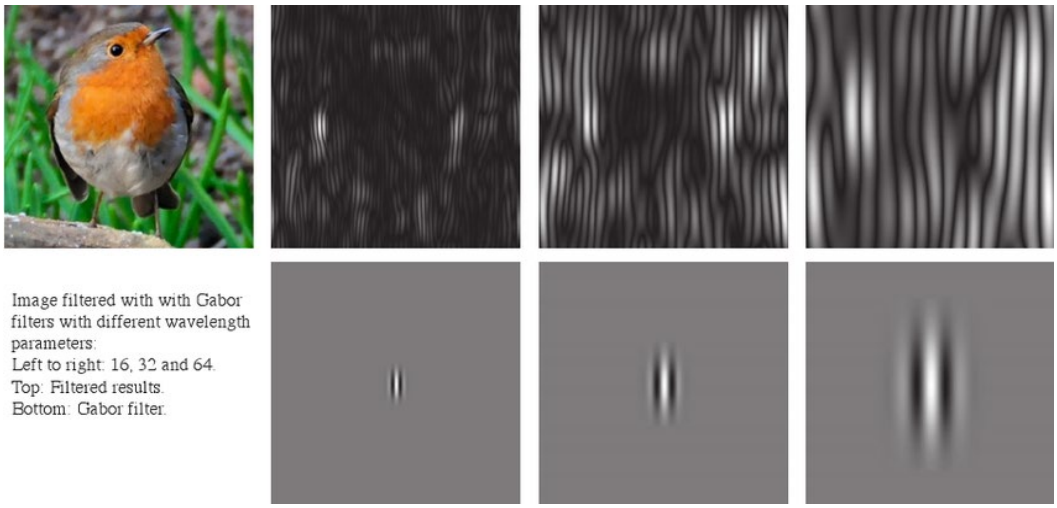


Figure 15: Gabor Filters

The second part of the feature extraction process was where the Gabor filters were made use of. This process involves filters that depend on varying angles to be initiated. In our case, theta, sigma, and lamda were given varying values. They then extract features relating to the textures and edges of the image based on the value of the filters by using the Gabor Kernel function to get a digital filter. Above is an example of three different digital filters of the same bird image to the right of the bird while the Gabor filter itself is described below each individual image.

After the process is complete, the features are saved in an array which is then reshaped to one vector so that it can be used by the random forest classifier as multiple dimensions in the array can cause an error. In the classification process, the classifier is then trained by first being declared and then fitted with the recently acquired features and the encoded labels to finally train the classifier to be used by the test directory. Validation, using a confusion matrix and classification graph was performed to visualize the classification so as to avoid any form of underfitting making sure the classifier is suitable enough to proceed to the next step.

- ## What parameters have you chosen for your selected classifiers?

For the convolutional neural network, we chosen specifics parameters for each layer. The parameters we have chosen are: Conv2D, MaxPooling2D, ReLU, Dense, Dropout and SoftMax.

The creation of a model happens in a lifecycle of 5 steps: defining, compiling, fitting, evaluating and predictions. The first thing we have decided before everything else was the use of TensorFlow or any other library and as well as using keras. It was an easy decision to stick to using this library and keras. Due to the ease of use, Api of keras which has been recently integrated to TensorFlow allowing for better support was the choice for this model. I Secondly, the next step was to configure the choice for the definition of model. In this case, we have chosen sequential. The sequential model Api is an easy to use and recommended model that allows the user to add the layers to the model one line at a time (https://www.facebook.com/MachineLearningMastery 2019). It allows full

customisation that can help us understand better and define the model from input to output (https://www.facebook.com/MachineLearningMastery 2019). The next step is to define the compilation of the model. In this stage, we must select our optimiser and loss function (https://www.facebook.com/MachineLearningMastery 2019). We must use the library to select a function for the model to compile with a chosen configuration which can prepare the data structures needed for an efficient use of the model that is being made (https://www.facebook.com/MachineLearningMastery 2019). In our case, we have chosen Adam as an optimiser and binary cross entropy for the loss function. We had chosen binary cross entropy as a loss function as we are making a binary classification model and used Adam for ease of use and online community support. After, we then have the fitting of the model. The fitting of a model is described as the selection of training configurations such as epoch and batch size (https://www.facebook.com/MachineLearningMastery 2019). Using the Api, we must call a function to do the training process based on the model and data set (https://www.facebook.com/MachineLearningMastery 2019). There are multiple reasoning and studies under the number suggestion of epoch and batch sizes. In our case, 15 epochs to try and make the model more accurate. Regarding the batch size, we had gone thru multiple examples of CNN binary classification and found that 100 to 150 is a good range therefore we had chosen 100. The next step is evaluation. This process we can unbiasedly check the performance of the model and use the library for functions to get metrics (https://www.facebook.com/MachineLearningMastery 2019). In our case, we have used classification report and confusion matrix. Finally, is the prediction where we use data the model has not seen before to make predictions based on the project.

For the model building, we have various layers connected to make the neural network. The first layer is the Conv2D which is the convolutional layer. It is a function part of TensorFlow used to create a feature map neural network and has variable parameters we tweaked accordingly. The other layer is the Max Pooling 2D layer which is responsible for selecting the maximum element of the feature map when covered by the filter. There are other options for this layer as well like Average pooling, but we decided to use Max pooling for ease of use and online support. The other layer present is the flatten layer. This layer is used to flatten the resulting array of images for classification purposes. Additionally, we have the dense layer which is a hidden layer with a parameter of how many neurons are present. Another important parameter in the model is the use of ReLU. ReLU is used as an activation here to help with the non-linearity.

For the Random Forest Classifier, only the n_estimators parameter was used. This parameter is used to determine exactly how many decision trees will be involved in the classification process. As the classifier uses multiple decision trees along with a voting system to choose the most accurate class, increasing the number of estimators results in higher accuracy when classifying the images in the test directory in the evaluation process. The accuracy was found to decrease if the parameter was given a high value that exceed the optimal value of decision trees. Through a trial-and-error process, a decision of 50 trees was reached as it produced the best results.

Evaluation

What is the model evaluation procedure?

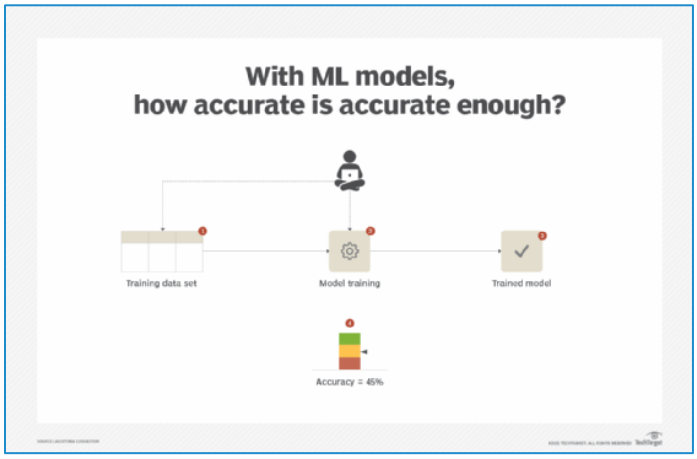


Figure 15: Model Evaluation

Model evaluation is the task of aiding human programmers to figure out which algorithm best suite a given problem and dataset (*Guide to Machine Learning Model Evaluation and its Techniques* 2021). 'Best Fit' is the term in machine learning that focuses on the task of evaluating the performance of different machine learning techniques using the same dataset and comparing the accuracy of the prediction in the outcome of each (*Guide to Machine Learning Model Evaluation and its Techniques* 2021). Better prediction is the metric that used in evaluating and comparison of various machine learning techniques and the highest accuracy in prediction is considered the best (*Guide to Machine Learning Model Evaluation and its Techniques* 2021).

There are two main techniques that are used to evaluate model performance: Holdout and Cross validation (*Guide to Machine Learning Model Evaluation and its Techniques* 2021). The holdout method is a technique used for finding the model performance and uses both testing and training data sets to calculate (*Guide to Machine Learning Model Evaluation and its Techniques* 2021). It works by checking how well the machine learning model developed using differing types of algorithm techniques used on never seen data samples (*Guide to Machine Learning Model Evaluation and its Techniques* 2021). This method allows a simple, flexible and fast approach (*Guide to Machine Learning Model Evaluation and its Techniques* 2021). The second technique is Cross validation where the process is involved dividing the dataset into samples and then checking the learning model with use of the other sample to figure out the accuracy of the model (*Guide to Machine Learning Model Evaluation and its Techniques* 2021). Additionally, there are 3 methods used for cross validation: validation, leave one out cross-validation (LOOCV) and K-fold Cross Validation (*Guide to Machine Learning Model Evaluation and its Techniques* 2021). It must be there are different types of predictive models that used to make prediction outcomes (*Guide to Machine Learning Model Evaluation and its Techniques* 2021). Many different metrics are possible such as classification accuracy, confusion matrix, logarithmic loss, area under curve anf –measure.

How are the results measured?

This process is highly important in determining the validity of your model; thus, the appropriate methods should be used to measure the results of the classification. Depending on the method chosen, critical details about a model can be pointed, such as the accuracy of the overall results as well as the level of bias & variance to reveal problems of underfitting or overfitting. The evaluation methods we ended up choosing were the classification graph and confusion matrix. We concluded that the confusion matrix will provide a direct and easy approach to figuring out exactly how many predictions were correct as compared to those that turned out to be wrong. On the other end, the classification report offers a better visualisation of the statistics offering values such as precision and f1 score to further understand the

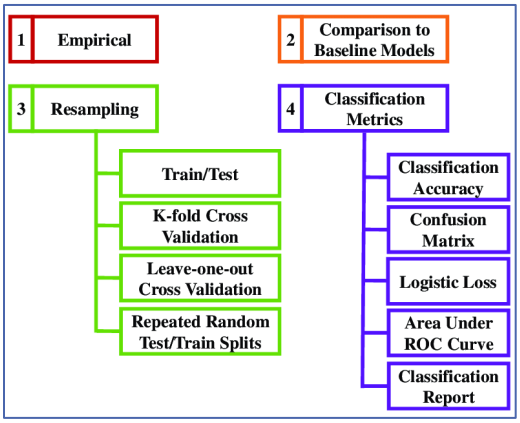


Figure 16: Different Methods of Evaluation

Confusion matrix

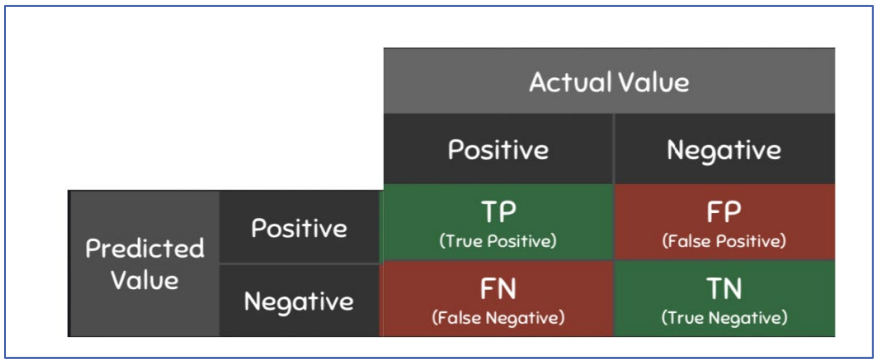


Figure 17: Binary Confusion Matrix

As shown in the image above, the confusion matrix was one of the methods chosen to evaluate our classification results. As the classifier only deals with two classes, it is considered binary and thus the matrix is shown to have only four cells. The four cells of the matrix are represented by TP, FP, FN, and TN. TP, also called True Positive is the cell which represents all the images that were classified or predicted as positive or recyclable in our case and were recyclable as well. Similarly, the True Negative cell contains the number of images that labelled as non-recyclable by both the classifier and the designated folders. The False cells are the opposite of each of the previously mentioned cells. False Positive is where the prediction was recyclable but was in fact the opposite of that. False Negative is where the image was predicted as non-recyclable, however the assigned folder was the opposite.

Our objective is to populate the True cells with the highest number possible indicating extremely accurate classification. A max value of 1600 is possible in each true cell. The matrix is quick and easy to use and although it provides us with a clear number of incorrect and correct predictions, it sometimes does not help us achieve the root of any accuracy issues.

Classification Report

	precision	recall	f1-score	support
0	0.77	0.86	0.81	37584
1	0.84	0.75	0.79	37577
accuracy			0.80	75161
macro avg	0.81	0.80	0.80	75161
weighted avg	0.81	0.80	0.80	75161

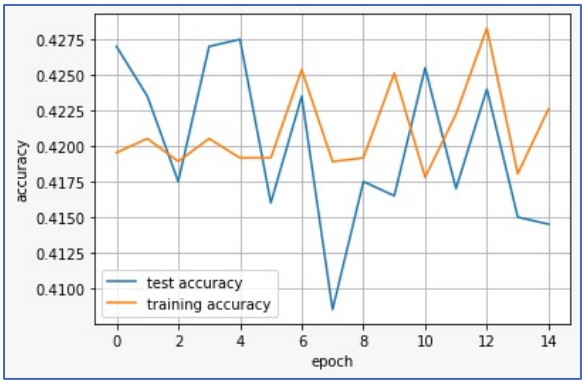
Figure 18: Classification Report Statistics

The report provides a larger variety of information regarding our classification processes. As mentioned before, precision, f1 score, along with recall & support as well are mentioned as seen in the image above. Precision refers to the ratio of true positive to the sum of the true positives and false positives. On the other hand, recall refers to another ratio, this time being that of the true positives to the sum of true positives and false negatives. The F1 score tells us the % of positive predictions that turned out to be correct. Finally, support just shows the actual occurrences a single class in the dataset being classified. These values provide an overall evaluation of the performance which helps us better understand our model and what it could be lacking or what it requires to be fixed.

- ## What are the results for each dataset and classifier?

Deep Learning

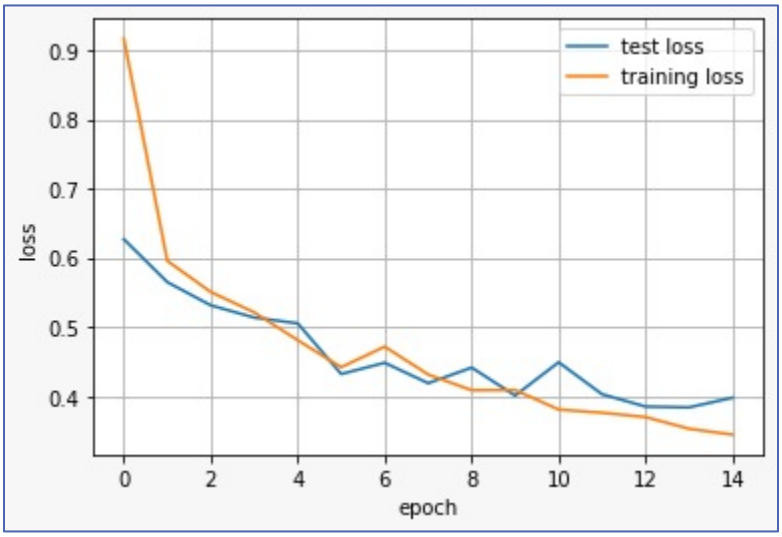
Accuracy




```
32/32 [==============================] - 9s 242ms/step - loss: 0.4035 - accuracy: 0.4209
Accuracy: 42.09%
```

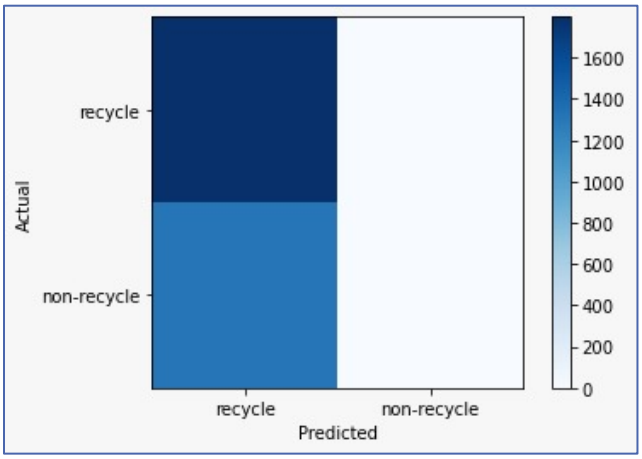
The first result to be discussed is accuracy and Val accuracy between test and training dataset. Across 15 epoch, there was an extreme fluctuation between the test and training data. But towards the end of 15 epochs, there was a difference with both image sets. The test accuracy is on the decrease while the training accuracy was increasing. It shows that there are still more features to learn.

Loss



The other result that needs to be discussed is the loss between the training and the test data. In this graph, we can see that the common trend is that its decreasing over the epoch. The loss and Val loss for both sets are on the same trend and are closely knitted towards lower loss.

Confusion Matrix



The evaluation method that also needs to be discussed is the confusion matrix for the model. Using this matrix, we can confirm that there were a lot of true positive and as false negatives. This also shows that there is high bias and low variance between the datasets.

Classification Report

Classification Report

	precision	recall	f1-score	support
0	0.58	1.00	0.73	1798
1	0.00	0.00	0.00	1307
accuracy			0.58	3105
macro avg	0.29	0.50	0.37	3105
weighted avg	0.34	0.58	0.42	3105

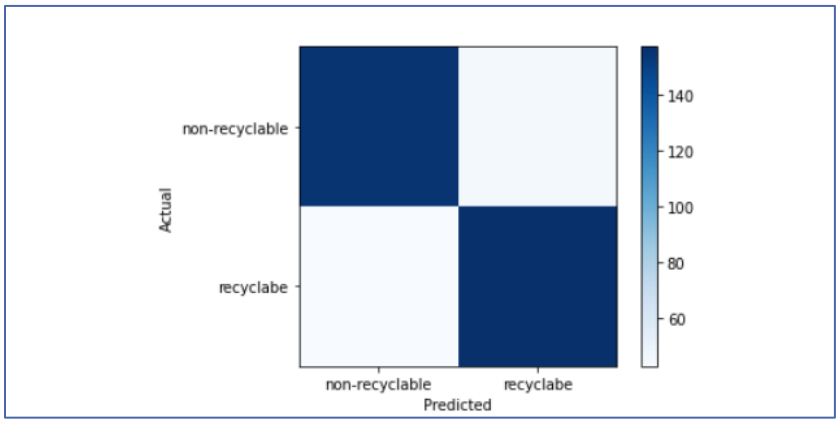
The classification report is another evaluation we can check the model. For the model, we can see that precision recall and f1-score are all 0. This shows that is still a lot more to learn and there is a bias in the model.

Traditional Machine Learning

Training Dataset

Confusion Matrix

```
Accuracy =  1.0
Confusion Matrix
[[1600    0]
 [   0 1600]]
```



The matrix in the training dataset indicates that the model has a 100 % accuracy with the recyclable images and non-recyclable images being classified correctly ten times out ten as indicated by the True Positive and True Negative cells being assigned the highest number while the other cells are equal to zero. The graph provides a more detailed view of the matrix.

Classification Report

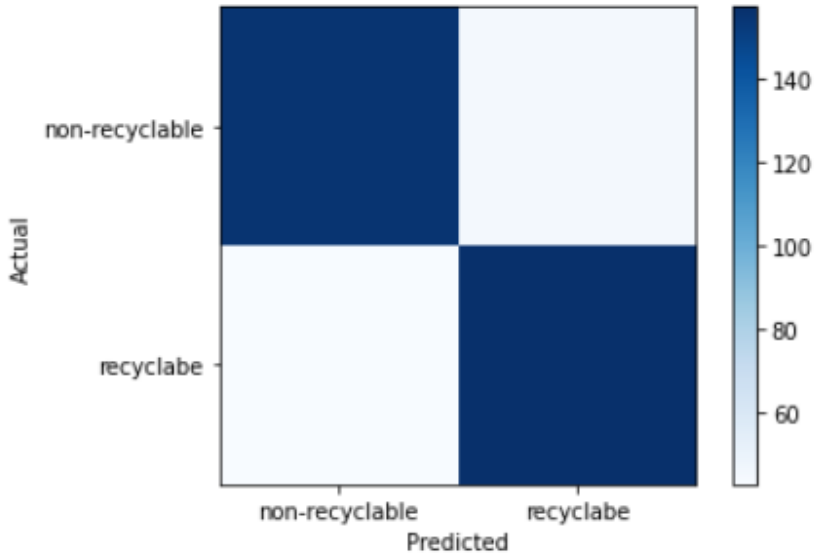
```
Classification Report
                precision    recall  f1-score   support

non-recyclable       1.00      1.00      1.00      1600
    recyclable       1.00      1.00      1.00      1600

      accuracy                           1.00      3200
     macro avg       1.00      1.00      1.00      3200
  weighted avg       1.00      1.00      1.00      3200
```

Like the confusion matrix, the classification report shows a 100 percent accuracy rate. The precision, recall, and f1 score values all amount to 1 this backing claim.

Test Dataset

Confusion Matrix

```
Accuracy =  0.78
Confusion Matrix
[[155  45]
 [ 43 157]]
```



The matrix in the test dataset indicates that the model has a 78 % accuracy with the recyclable images and non-recyclable images being classified correctly most of the time. This is indicated by the cells of the matrix where non-recyclable images are correctly predicted 155 times out the total 200 pictures and the recyclable images are given the right prediction 157 times. The graph provides a more detailed view of the matrix.

Classification Report

Classification Report	precision	recall	f1-score	support
non-recyclable	0.78	0.78	0.78	200
recyclable	0.78	0.79	0.78	200
accuracy			0.78	400
macro avg	0.78	0.78	0.78	400
weighted avg	0.78	0.78	0.78	400

In this classification report, we can see that the f1 score, precision, and recall values back the confusion matrix by showing a result of or close to 0.78.

Discussion

Comment on the overall results obtained. Are the results appropriate, does the classifier work, etc.

In the case of the machine learning model, the results obtained were adequate. Firstly, feature extraction was done through collecting the pixel sizes and via the Gobar filters acquiring features related to textures. The Random Forrest classifier was then used and trained to create a functioning image classification model. This helped us obtain perfect values for the training data set. The confusion matrix displayed values only in the True Positive and True Negative Cells indicating 100 % accuracy. While, the classification report had precision, recall, and f1 score values all of 1 which once again proved that at least for the training data set, the classifier worked well enough and was trained enough to not show any problems relating to underfitting. This gave us the necessary validation that was required to move on to the next step of evaluation of the test directory where the classifier encounters images previously not seen.

Through the repeated use of the confusion matrix and classification graph, we realized that for test directory, the images were not classified all of time, but the procedure was still mostly accurate with a rate of 78 %. Out of the 200 images of each class, more than a 150 of them were correctly labelled every time. The report displays a similar outcome with the values of precision, f1 score, and recall, all being around 0.78. This rate of accuracy is further backed by a random image chooser from the test directory which displays the actual and predicted labels of the image. The test showed that 3 out of four images randomized were correctly labelled. To summarize, the classifier works accurately for large percentage of the test directory and thus could be used in a real-life situation, if the accuracy can be improved by using a larger training data set. Further varying Gabor filters can also be utilised to contain more features for improving the accuracy rate to possibly 95 to 100.

As for the neural network, I think that the results showed that it needs more data to work. It shows that there is high bias. Even though it can classify, it is incomplete and needs more data and epoch to learn more about the data.

- ## What are the limitations of your approach? Under what conditions will it fail? Under what conditions will it work better?

  The obvious limitation for the traditional machine learning approach is the size of the dataset. A smaller sized dataset is not able to provide a good number of features resulting in a poor accuracy rate. We used 3200 images for our entire training dataset to achieve a good accuracy rate, however the number of images had to be reduced initially from around 13000. We require images in our training set to be at least around 12000 to 20000 because a lot of the images such as trash, biological, and plastic are extremely broad and require a high number of images to account for this. However, such a thing could not be accomplished due to the weak nature of the operating environment.

  Additionally, multiple colour channels are used to get features for an image, this results in the need for higher computational power as the complexity of the procedure much higher than just using grayscale. Thus, low-end system will have longer time computing, or they may even crash.

  The limitation if the CNN approach is the hardware. We have an issue regarding epochs and dataset and therefore we cannot get accurate results. The condition will fail if there if there is further lessening of data and processing for the neural network.

- ## Can the classifier be improved? Suggest possible improvements.

  The Random Forest classifier can be improved in a variety of ways. The first one being the addition of data. Various images in dataset are broadly termed such as trash and biological, which is why images to up to at least 12000 to 20000 are required. A better operating environment can allow us to do such a thing. On the other hand, the number of features being collected can be increased. One such feature could be adding further variations into the Gabor filters such as varying the value gamma instead of making it consistent as before. Another feature collection can be the use of Sobel to detect edges thus further increasing the number of features. The image can also be resized to a larger version to allow the system to possibly detect more distinctive features and improving the probability of achieving a 100 % rate of accuracy. Finally, transfer learning can be used to obtain pre-existing and trained models. These models are extremely accurate and can detect a variety of features of an image, thus allowing them to easily distinguish between the various images.

  For the neural network, more layers can be added. We can add more data sets and more layers and do more epochs. We can further improve by playing around with epoch and batch size for better results. Overall, more layers, datasets and epochs should improve the neural network.

- ## Which part of the machine learning process flow can be optimized to improve results?

  The learning algorithm part of the learning process can be optimized to improve the results. One such way is the use of hyperparameter optimization. This finds the most optimal hyperparameter of the model. Hyper parameters are set by the machine learning engineer and include examples such as the number of decision trees in random forest as well as the k in k-nearest neighbours.

  ### *Are there any differences if you choose more or less classes?*

  In our opinion, having more classes will require more dataset and a complete change in architecture. Optimising a result will have to look for more other ways to optimise such as adding layers or using more neurons.

Are there any differences if you pre-process the data?

If the dataset was further pre-processed, it should increase the accuracy and efficiency of the machine learning model.

Conclusion

After having performed the deep learning algorithm as well as the traditional machine learning one, we have not only learned of the different ways image classification is achieved but as well as the differences in effectiveness and efficiency of each type. For the deep learning algorithm, we found it to more complex to work on than the traditional one. This is due to the reason that no features are extracted, thus the model must figure out everything itself. For this reason, it also takes a way longer time to achieve. Its complexity also requires the operating environment to be relatively advanced so it could deal with much more information. For these reasons, it was found to be harder to setup and maintain. The accuracy through evaluation was thus found to be just a dismal 42%. On the other hand, the traditional machine learning was much easier to deal with, it involves feature extraction and classification separately. Because the features are specified, it easier to classify through the chosen random forest classifier. The requirements for the operating environment are not as taxing and allow for weaker systems to use it. The processing time for the training is also more efficient. Overall, it's easier to understand and thus optimize, which resulted in a 78 % rate of accuracy. One major advantage is because, due to feature extraction process, it requires relatively a lot less images than the deep learning algorithm, thus stressing the need for an advanced operating environment. For these reasons, we conclude that the broad non deep learning method is easier to use and efficient and this more suitable.

References

Find Good Data Sets 2022, Tableau.com, viewed 20 May 2022, <https://help.tableau.com/current/pro/desktop/en-us/find_good_datasets.htm>.

How to Ensure Image Dataset Quality for Image Classification? 2022, Picsellia.com, viewed 20 May 2022, <https://www.picsellia.com/post/image-data-quality-for-image-classification>.

How to Build a Dataset for Image Classification 2022, Levity.ai, viewed 20 May 2022, <https://levity.ai/blog/create-image-classification-dataset>.

Rosebrock, A 2019, *Keras ImageDataGenerator and Data Augmentation - PyImageSearch*, PyImageSearch, viewed 21 May 2022, <https://pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/>.

Hackl, C 2020, 'How 4 Companies Are Using AI To Solve Waste Issues On Earth And In Space', *Forbes*, 21 July, viewed 21 May 2022, <https://www.forbes.com/sites/cathyhackl/2020/07/18/how-4-companies-are-using-ai-to-solve-waste-issues-on-earth--in-space/?sh=16fd0eab35fa>.

Michael 2021, *How Artificial Intelligence Is Transforming Waste Management*, Keymakr's Blog features the latest news and updates, Keymakr's Blog features the latest news and updates, viewed 21 May 2022, <https://keymakr.com/blog/how-ai-is-transforming-waste-management/>.

Fumo, D 2017, *Types of Machine Learning Algorithms You Should Know*, Medium, Towards Data Science, viewed 21 May 2022, <https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861>.

Wakefield, K 2022?, 'A guide to the types of machine learning algorithms and their applications', SAS UK, viewed 22 May 2022, <https://www.sas.com/en_gb/insights/articles/analytics/machine-learning-algorithms.html>.

Mesevage, T G 2020, 'Machine Learning Classifiers - The Algorithms & How They Work', MonekyLearn, viewed 22 May 2022, < https://monkeylearn.com/blog/what-is-a-classifier/ >.

Simran Bansari 2019, *Introduction to how CNNs Work - DataDrivenInvestor*, Medium, DataDrivenInvestor, viewed 22 May 2022, <https://medium.datadriveninvestor.com/introduction-to-how-cnns-work-77e0e4cde99b>.

https://www.facebook.com/MachineLearningMastery 2019, *TensorFlow 2 Tutorial: Get Started in Deep Learning With tf.keras*, Machine Learning Mastery, viewed 23 May 2022, <https://machinelearningmastery.com/tensorflow-tutorial-deep-learning-with-tf-keras/>.

Guide to Machine Learning Model Evaluation and its Techniques 2021, Knowledgehut.com, viewed 23 May 2022, <https://www.knowledgehut.com/blog/data-science/machine-learning-model-evaluation>.

Appendix

CNN

```
#!/usr/bin/env python
# coding: utf-8

# <h1 color='black' size="20">Introduction to Artificial Intelligence iRobot
Assignment 2 <br>Convolutional Neural Network</font>

# <h2 color='black' size="20">Importing Libraries</font>
#

# In[1]:


import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential, load_model
from keras.layers import Dense, Dropout, Activation, Flatten, Conv2D,
MaxPooling2D
from sklearn.metrics import confusion_matrix, classification_report


# <h2 color='black' size="20">Data Path and Image Processing</font>

# In[4]:


#Image Pre-processing
#variables

#defines the image height, width and channel
img_height, img_width, img_ch = 256, 256, 3

#defines the batch size
batch_size =100

#epochs - number of loops thru training set
epochs=2

#file path from directory
dir_tr ='Data/train/'
dir_test ='Data/test/'
```

```
#imagedatagenrator from tensorflow - real time augmentation of images (train)
train_datagen = ImageDataGenerator(
        #rescale the image sizes with ration 0 to 1 from 0 to 255
        rescale=1./255,
        #slant the images
        shear_range=0.2,
        #zoom the images
        zoom_range=0.2,
        #flip the image horizontally
        horizontal_flip=True)

#imagedatagenrator from tensorflow - real time augmentation of images (test)
test_datagen = ImageDataGenerator(rescale=1./255) #scale the images

#saving to a variable the loading of the image dataset in memory and creating
batches
#of augmented images from file path
train_generator = train_datagen.flow_from_directory(
        #the directory path of the training images
        dir_tr,
        #target size of what dimension the images are resized
        target_size=(img_height, img_width),
        #batch sie - number of images given in 1 iteration
        batch_size=batch_size,
        #mode of label array - binary means 1d arrary labels
        class_mode='binary')

#saving to a variable the loading of the image dataset in memory and creating
batches
#of augmented images from file path
test_generator = test_datagen.flow_from_directory(
        #the directory path of the test images
        dir_test,
        #target size of what dimension the images are resized
        target_size=(img_height, img_width),
        #batch sie - number of images given in 1 iteration
        batch_size=batch_size,
        #mode of label array - binary means 1d arrary labels
        class_mode='binary')

#puts into a variable the images followed by the labels with next to move next
into the array
images, labels = next(train_generator)
```

```
#print out the labels - 0 means non recycable & 1 means recyclable
train_generator.class_indices


# <h2 color='black' size="20">Display Dataset</font>

# In[5]:


#using matplotlib create a figure object
plt.figure(figsize=(20, 20))#configure the figure size to a width of 20 and
height of 20

#display the image sets from the train data et
n = 5  #how any plots will be displayed
#using a forloop for the amount of plots
for i in range(n):
    # display original
    ax = plt.subplot(2,n, i + 1)
    #display the data as an image based on the forloop
    plt.imshow(images[i])
    #show the axis
    ax.get_xaxis().set_visible(True)
    ax.get_yaxis().set_visible(True)

plt.show()


#  <h2 color='black' size="20">Building and Compiling Model</font>

# In[14]:


#Code for the model and the layers

#sequantial model api -simplest and allows us to define the layers in a linear
order from input to output
model = Sequential()
#conv2D - neural networks create a feature map (convolutional layer)
model.add(Conv2D(64, (3, 3), padding='same', activation = 'relu', input_shape =
(img_height, img_width, img_ch)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))

#conv2D - neural networks create a feature map (convolutional layer)
model.add(Conv2D(64, (3, 3), activation='relu'))
```

```
#MaxPooling2D - selects the maximum element from the region of the feature map
covered by the filter
model.add(MaxPooling2D(pool_size=(2, 2)))
#Dropout - drops some of the neruons
model.add(Dropout(0.4))

#conv2D - neural networks create a feature map (convolutional layer)
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))

# This layer flattens the resulting image array to 1D array
model.add(Flatten())

# Hidden layer with 64 neurons and Rectified Linear Unit activation function
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(1))

# Output layer with single neuron using softmax
model.add(Activation('softmax'))

model.summary()
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =
['accuracy'])


# <h2 color='black' size="20">Model Training</font>

# In[15]:


history = model.fit_generator(
        train_generator,
        steps_per_epoch=((train_generator.samples)//150),
        epochs=15,
        verbose=1,
        validation_data=test_generator,
        validation_steps= ((test_generator.samples)//150))


# <h2 color='black' size="20">Model Evaluation Graphs</font>

# In[16]:
```

```
############Plot Accuracy and Loss over Epochs######
plt.plot(history.epoch,history.history['val_accuracy'],label='test accuracy')
plt.plot(history.epoch,history.history['accuracy'],label='training accuracy')
plt.legend(loc=0)
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.grid(True)
plt.show()

#############
plt.plot(history.epoch,history.history['val_loss'],label='test loss')
plt.plot(history.epoch,history.history['loss'],label='training loss')
plt.legend(loc=0)
plt.xlabel('epoch')
plt.ylabel('loss')
plt.grid(True)
plt.show()


# In[17]:


scores = model.evaluate(test_generator, verbose=1)
print("Accuracy: %.2f%%" % (scores[1]*100))


# <h2 color='black' size="20">Save Model into a file</font>

# In[18]:


model.save('model.h5')


# <h2 color='black' size="20">Load Model from file</font>

# In[13]:


model = load_model('model.h5')

model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

```
# <h2 color='black' size="20">Prediction with model</font>

# In[14]:


img = image.load_img('test1.jpg', target_size=(img_height, img_width))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)

images = np.vstack([x])
classes = np.argmax(model.predict(x), axis=-1)
print(classes)


# <h2 color='black' size="20">Evaluation</font>

# In[19]:


classes=['recycle','non-recycle']

Y_pred = model.predict(test_generator, steps = np.ceil(test_generator.samples /
test_generator.batch_size), verbose=1, workers=0)
y_pred = np.argmax(Y_pred, axis=1)

cm = confusion_matrix(test_generator.classes, y_pred)

#prints confusion matrix
print('Confusion Matrix')
print(cm)

#prints classification report
print('\nClassification Report')
print(classification_report(test_generator.classes,y_pred))

plt.imshow(cm, cmap=plt.cm.Blues)
plt.xlabel("Predicted")
plt.ylabel("Actual")
ticks = np.arange(len(classes))
plt.xticks(ticks, classes)
plt.yticks(ticks, classes)
plt.colorbar()
plt.show()
```

```
# In[ ]:
```

Traditional Machine Learning

#import libraries

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
import cv2
import seaborn as sb
from sklearn import metrics
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
import numpy as np
import mahotas
import pandas as pd
import glob
import os
from skimage.filters import sobel
import h5py
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plot

#preprocess images (resize)

def imageProcess(dir,images,classes):

    for path in glob.glob(dir):
```

```
        class1 = path.split("\\")[-1]

        for img_path in glob.glob(os.path.join(path, "*.jpg")):
            img = cv2.imread(img_path, cv2.IMREAD_COLOR)
            img = cv2.resize(img, (128, 128))
            images.append(img)
            classes.append(class1)

images_train=[]
classes_train=[]
classes_train_encoded=[]

images_test=[]
classes_test=[]
classes_test_encoded=[]

imageProcess("images/train/*",images_train,classes_train)
imageProcess("images/test/*",images_test,classes_test)



#encode labels to be understood by computer
encode = LabelEncoder()
encode.fit(classes_train)
classes_train_encoded = encode.transform(classes_train)
encode.fit(classes_test)
classes_test_encoded = encode.transform(classes_test)

#scale image values
images_test= [x / 255.0 for x in images_test]
images_train= [x / 255.0 for x in images_train]

images_train = np.array(images_train)
```

```
classes_train = np.array(classes_train)
images_test = np.array(images_test)
classes_test = np.array(classes_test)

def feature_extraction(images):

    filtered_features = pd.DataFrame()
    im=images
    im=images * 255.0

    for img in range(images.shape[0]):

        features = pd.DataFrame()
        img_input = images[img, :,:,:]

         # 1 - Pixel values

        #Add pixel values to the data frame
        pixel_values = img_input.reshape(-1)
        features['Pixel_Value'] = pixel_values



        # 2 - Gabor filter responses

        num = 1
        kernels = []
        for theta in range(2):
            theta = theta / 4. * np.pi
            for sigma in (1, 3):
                lamda = np.pi/4
                gamma = 0.5
                gabor_label = 'Gabor' + str(num)
```

```
            print(gabor_label)
            ksize=9
            kernel = cv2.getGaborKernel((ksize, ksize), sigma, theta, lamda, gamma, 0, ktype=cv2.CV_32F)
            kernels.append(kernel)

            fimg = cv2.filter2D(img_input, cv2.CV_8UC3, kernel)
            filtered_img = fimg.reshape(-1)
            features[gabor_label] = filtered_img
            print(gabor_label, ': theta=', theta, ': sigma=', sigma, ': lamda=', lamda, ': gamma=', gamma)
            num += 1



        print(img)

        filtered_features = filtered_features.append(features)

    return filtered_features

#extract features
train_features=feature_extraction(images_train)

n_features = train_features.shape[1]
train_features = np.expand_dims(train_features, axis=0)
X_for_RF = np.reshape(train_features, (images_train.shape[0], -1))

#train
clf  = RandomForestClassifier(n_estimators=50)
clf.fit(X_for_RF, classes_train_encoded)
clf_pred = clf.predict(X_for_RF)

#check validity of classifier (training set)
test_prediction1 = clf.predict(X_for_RF)
```

```
test_prediction1 = encode.inverse_transform(test_prediction1)
print ("Accuracy = ", metrics.accuracy_score(classes_train, test_prediction1))
cm = confusion_matrix(classes_train, test_prediction1)
print('Confusion Matrix')
print(cm)

print('\nClassification Report')
print(classification_report(classes_train,test_prediction1))

classes=['non-recyclable','recyclabe']

plot.imshow(cm, cmap=plot.cm.Blues)
plot.xlabel("Predicted")
plot.ylabel("Actual")
ticks = np.arange(len(classes))
plot.xticks(ticks, classes)
plot.yticks(ticks, classes)
plot.colorbar()
plot.show()

#predict the test dir
test_features = feature_extraction(images_test)
test_features = np.expand_dims(test_features, axis=0)
test_for_RF = np.reshape(test_features, (images_test.shape[0], -1))

#check validity of classifier (test set) -confusion matrix
test_prediction = clf.predict(test_for_RF)
test_prediction = encode.inverse_transform(test_prediction)
print ("Accuracy = ", metrics.accuracy_score(classes_test, test_prediction))
cm = confusion_matrix(classes_test, test_prediction)

print('Confusion Matrix')
```

```
print(cm)

#check validity of classifier (test set) -classification graph

print('\nClassification Report')
print(classification_report(classes_test,test_prediction))

plot.imshow(cm, cmap=plot.cm.Blues)
plot.xlabel("Predicted")
plot.ylabel("Actual")
ticks = np.arange(len(classes))
plot.xticks(ticks, classes)
plot.yticks(ticks, classes)
plot.colorbar()
plot.show()

#test random image
import random
n=random.randint(0, images_test.shape[0]-1) #Select the index of image to be loaded for testing
img = images_test[n]
plot.imshow(img)

#Extract features and reshape to right dimensions
input_img = np.expand_dims(img, axis=0) #Expand dims so the input is (num images, x, y, c)
input_img_features=feature_extraction(input_img)
input_img_features = np.expand_dims(input_img_features, axis=0)
input_img_for_RF = np.reshape(input_img_features, (input_img.shape[0], -1))
#Prediqct
img_prediction = clf.predict(input_img_for_RF)
img_prediction = encode.inverse_transform(img_prediction)  #Reverse the label encoder to original name
print("The prediction for this image is: ", img_prediction)
print("The actual label for this image is: ", classes_test[n])
```